# In-network Solution for Network Traffic Reduction in Industrial Data Communication

Csaba Györgyi*, Károly Kecskeméti*, Péter Vörös*, Géza Szabó† and Sándor Laki*
*ELTE Eötvös Loránd University, Budapest, Hungary
†Ericsson Research, Budapest, Hungary

***Abstract*—Industrial networks rely on standard real-time communication protocols like ProfiNet. These protocols are used for cyclic data exchange between IO devices and controllers. Since continuous monitoring of IO devices is important, a large number of data packets can be observed in such field networks between the IO devices and controllers. Each IO device cyclically reports its data or internal state to a controller at a predefined frequency. However, the reported data of most IO devices are not changing all the time, and thus the same bytes are transmitted multiple times. The majority of data packets is only used for checking the availability of such devices. In this paper, we consider an industrial environment where IO devices are located in an industrial site while controllers are running remotely (e.g., a software PLC in a private or edge cloud), and there is a radio link (e.g., 5G radio) between the two sides. We propose an in-network traffic reduction method that filters out the unnecessary data traffic at the two ends of the radio link, detects failure of devices and the radio link fast, and does not require any modification in the IO devices and controllers. Our solution is based on the cooperation of two P4-programmable networking elements deployed at the two sides of the radio link. Our preliminary measurements with P4-programmable hardware switches and emulated ProfiNet devices show that the method can significantly reduce the load on the radio link, while it could seamlessly be deployed in existing industrial environments.**

***Index Terms*—traffic reduction, in-network computing, P4, ProfiNet, industrial data communication**

## I. Introduction

Industrial networks evolve rapidly. Cloud-based software solutions have emerged to replace industrial controller hardware. With the advent of 5G, stable network connectivity with high reliability can also be established over radio between different parts of the industrial site or other remote locations (e.g., private and edge cloud nodes). However, connecting programmable logic controllers (PLCs) and IO devices via 5G radio presents a number of new challenges. IO devices and PLCs continuously send status signals with predetermined frequencies (1 ms to a few seconds). Most of these messages can be considered as simple life-signals, repeating data already transferred. This eventually results in a large number of small packets to be transmitted over the radio link, leading to high overhead affecting both spectral and energy efficiency. Both PLCs and IO devices are sensitive to packet loss and jitter, tolerating 2-3 missing data packets in a connection.

In this paper, we focus on a specific industrial protocol called ProfiNet [1] that implements a communication behavior called cyclic data exchange. Accordingly, each IO device and the appropriate PLC cyclically exchange data packets with a predefined frequency. We propose a new method using the concept of in-network computing to substantially reduce the network traffic on the radio link by getting rid of unnecessary data transmission from IO devices whose internal data/state remained unchanged. Our solution does not require any changes in the existing protocol used by both PLC and IO devices. In addition, seamless deployment of this system is possible by placing two instances of the P4-programmable switches (or smartNICs) at the two sides of the critical link in the forwarding path. No further changes in the infrastructure or modification in the configurations are required. The proposed solution provides fast reaction to link and device failures, but still do not introduce a significant computational and memory overhead. The network load reduction achieved by eliminating redundant traffic greatly increases the reliability of communication.

The rest of the paper is organized as follows. First, we show an overview of the current state of the art in Sec. II, then we present an overview of our proof of concept system in Sec. III. It is followed by the implementation details in Sec. IV. In Sec. V, our preliminary evaluation using a hardware P4-Switch is shown. Finally, Sec. VI concludes our work.

## II. Related Work

The tendency of softwarization has reached the industrial manufacturing sector as well. Unfortunately, to get to a state in which any industrial communication protocol is easy to implement proves to be a slow process, since the industry has strict requirements regarding availability, security, and most importantly there is a high cost of acquiring devices.

ProfiNet is a standard aiming in the described direction, based on industrial Ethernet, providing numerous advantages: 1) It can simply wrap non-industrial and non-ProfiNet traffic, meaning a much wider range of devices that can be integrated. 2) Automatic discovery and classification of devices are possible. 3) Headers provide a straightforward method of differentiation between time-sensitive and regular packets.

To provide computer networks with a high degree of flexibility and scalability Software Defined Networking (SDN) [2] introduced a new way of programming abstractions by decoupling the data and the control plane functionality. While the literature of control plane programmability has a rich past, difficulties of programmable and portable data planes have just started gaining attention in recent years. For offering network

978-1-6654-0522-5/21/$31.00 ©2021 IEEE

developers the desired flexibility, specific programming languages have developed. These languages let experts describe the entire packet processing pipeline in a protocol-independent way from a high-level abstraction. P4 [3] is one of the language propositions, which has achieved the most influential community support, backed by members from both industry and academia. The language has numerous compilers for diverse software and hardware targets, ranging from general-purpose processors, NetFPGAs [4] and SmartNICs, to custom-designed sets of ASICs such as Intel Tofino.

Protocol independent network programming opens up the fields for a new era, in which switches are more than packet forwarding tools. Network hardware can also take part in calculations on the application level during the communication. This new paradigm is called in-network computing, where server-based computations are offloaded partly or completely to the programmable switches, such as in-band network telemetry [5], in-network caching, L7 load balancing, in-network key-value stores, AQM algorithms [6]. In-network computing grants an ultra-low latency and significantly higher throughput compared to its conventional equivalents.

## III. System Overview

A possible deployment scenario of a cloud-based industrial control is depicted in Figure 1. This figure illustrates the traditional communication pattern (red curve) between IO devices working in an industrial site and software PLCs running in the cloud (public, private, or edge). We assume that the industrial site has a 5G radio access, through which the soft-PLC is accessible. Observe that all the ProfiNet traffic goes through the radio link. Each PLC continuously queries the state of the IO devices with predefined frequencies. The update period is typically in a range of 1ms-10s and varies from device to device. In many cases, the reported states are identical, generating unnecessary traffic to be transmitted over the radio link. If the number of IO devices is large, the large number of small packets will pollute the radio link, degrading the radio performance and resulting in significant power consumption.

To avoid the transmission of unnecessary ProfiNet packets over the radio link, we propose an in-network traffic reduction method. Accordingly, we assume two P4-programmable switches at the two sides of the radio link (as shown in Fig. 2). The two programmable switches cooperate to keep track of the latest state (reported packet data) of ProfiNet devices, recognize device and radio link failures while minimizing the number of Profinet packets transmitted over the radio. As illustrated in the figure, each P4-switch emulates the presence of the devices (either IO devices or PLCs) located on the other side of the radio link, responding to Profinet packets of the local (deployed on the same side of the radio) devices.

Both P4-switches implement the same data and control planes, and no modification in the ProfiNet devices is needed.

## IV. Implementation

The proposed approach combines two key functionalities: 1) algorithms for storing and updating packet data and reducing

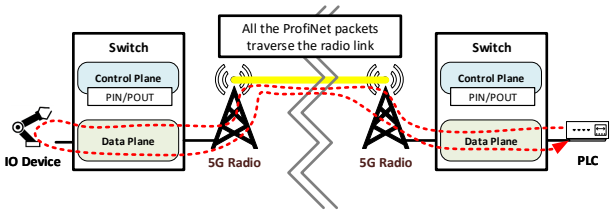


Fig. 1. Traditional communication between a PLC and an IO device over a radio access

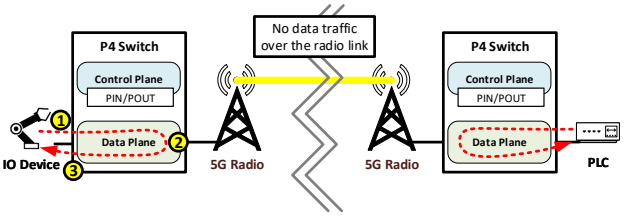


Fig. 2. The latest packet data is stored by the switch that is used to generate the expected reply of the other end.

data traffic over the radio link, and 2) Methods for monitoring the status of the radio link and the different ProfiNet devices. Though we discuss these functions separately, they are implemented as a single pipeline in our prototype.

### A. Data Traffic Reduction

In ProfiNet, an IO device and the responsible PLC continuously exchange state information encoded as IO data in the packet. The packet structure is the same in both directions. Generally, the devices send data packets with a predefined frequency and expect appropriate replies. The applied sending frequency is negotiated between the PLC and the IO device in advance, using control packets. In this section, we only focus on how the IO data traffic during operation can be reduced on the interconnecting radio link.

To this end, the pipeline has two main tasks: 1) handling incoming data packets from local ProfiNet devices and replying to them on behalf of other ProfiNet entities located on the other side of the radio link, without transmitting data to the other side, and 2) recognizing the change in the reported device state (IO data) and notifying the P4-switch on the other side of the radio link where the stored state information needs to be updated.

Fig. 2 depicts the key packet processing steps on how an incoming ProfiNet packet is handled: 1) A ProfiNet source *src*

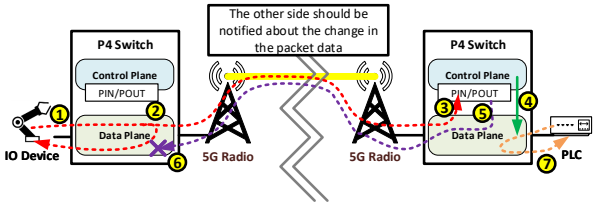


Fig. 3. If packet data of a given device is changed, the packet is forwarded to the switch on the other side so that it should update the stored data content.

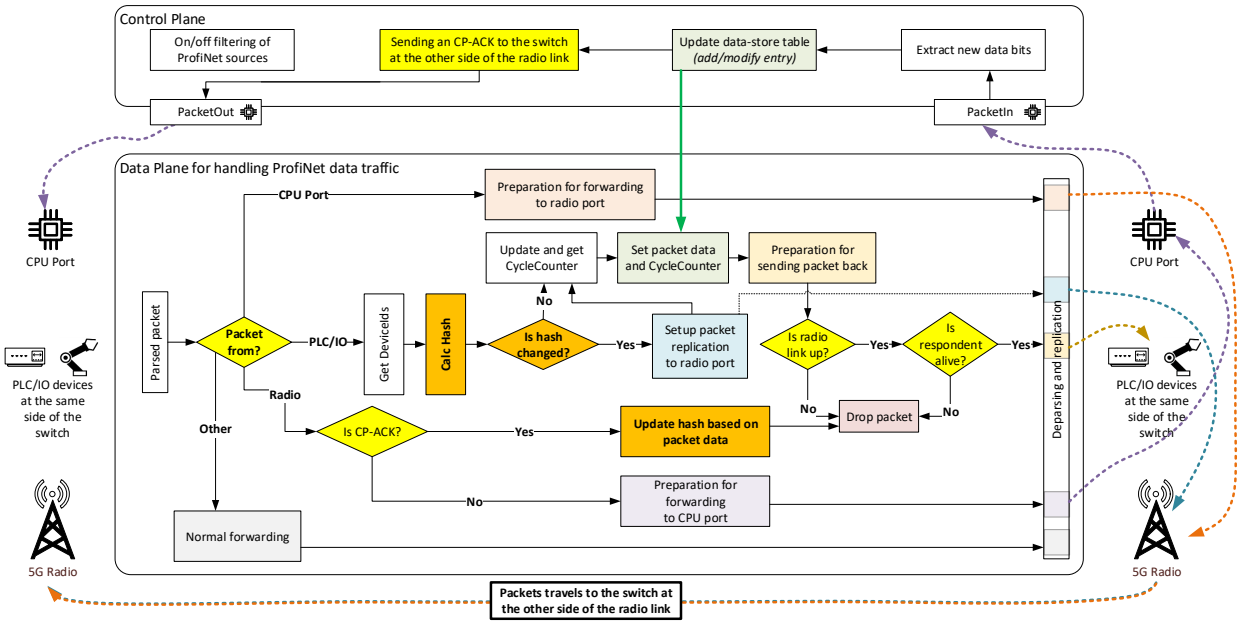


Fig. 4. Overview of the proposed ProfiNet IO data packet processing pipeline.

(the IO device in the figure) generates a packet to the ProfiNet destination *dst* (the PLC in the figure). The packet arrives at the P4-Switch deployed before the radio link. 2) As shown in Fig. 4, the switch first calculates a hash from the packet data. If it is the same as the value stored in a register for device *src*, it turns the packet back instead of forwarding through the radio link, update the packet as if it had been sent by device *dst*. This includes the modification of fields of the ProfiNet header: device identifier is set to the one of *dst*, cycle-counter is updated from a register, and IO data is filled with last seen bytes of *dst* from a match-action table. 3) The expected reply packet is sent to *src*. The device *src* will not be able to detect that it was not sent by *dst*.

Fig. 3 depicts the case when the packet data is changed: 1) As previously, ProfiNet source *src* generates a packet to the ProfiNet destination *dst*. 2) The P4-Switch before the radio link hashes the packet data and recognize its changed state. Then the packet is replicated and a copy of the original one is forwarded through the radio link, while the packet under processing is turned back to *src*, and updated as if it had been sent by device *dst* (see the previous case). 3) The copy of the original packet then arrives at the P4-switch on the other side of the radio link where the data plane forwards it to the control plane. 4) Based on the carried packet data, the control plane (CP) fills the match-action table responsible for storing the packet data of ProfiNet devices with the new data content of *src*. 5) The CP then sends the packet back to the data plane. The packets coming from the CP are marked with a CP-ACK flag and directed towards the radio link. 6) The packet flagged with CP-ACK arrives back to the P4-Switch at the *src* side, where the stored hash value of *src* is recomputed and updated according to the packet data (stored in a register). The CP-ACK packet is then dropped.

The full pipeline of both main tasks described previously is shown in Fig. 4. The pipeline uses two register arrays indexed by device identifiers that store the CycleCounters that should be shifted when a reply message is assembled, and the hash values used for checking the validity of packet data. The packet data is stored in a match-action table where the key is the device identifier and the action fills the IO data bytes of the ProfiNet packet with the latest observed values.

### B. Radio and Device Failure Detection

To make the application of the proposed data reduction method transparent to ProfiNet devices, it is important to react to failures as soon as possible. To this end, the proposed pipeline contains a mechanism that exchanges information about the availability of ProfiNet devices and the state of the radio link. If failure is detected, the automatic responses described in Sec. IV-A need to be stopped by the appropriate P4-Switch to ensure the original behavior of the industrial network. Each received ProfiNet packet sets a bit in a bitmap register expressing the aliveness of the sender device. The continuous reinforcement of this status bit is needed. If the device becomes unavailable, the P4-Switches stop transmitting automatic responses.

The pipeline part is summarized in Fig. 6 and Fig. 5. The main operational steps are the following: 1) A packet generator engine (hardware or software) generates a probe packet at a predefined frequency. 2) The probe is then filled with the status bitmap of known ProfiNet devices by the data plane. This bitmap carries information about the aliveness of devices at the local side of the radio link. To monitor the radio state both P4-Switches maintain a radio time-to-live (TTL) counter that is decremented by 1 whenever a probe is sent. If it reaches zero, the radio becomes unavailable. 3) The probe is sent over

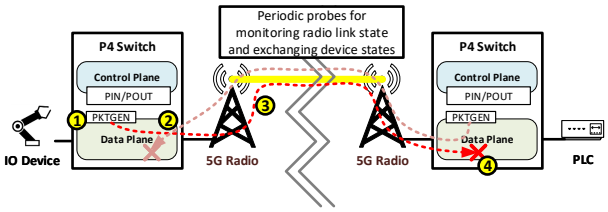


Fig. 5. Both switches periodically probe the connectivity between them and exchange information on device states.

the radio link. 4) If a probe is received from the other side of the radio link, the radio TTL in the recipient is set to the initial value (3 in our case). The bitmap of device states is refreshed according to the received information, and the probe packet is dropped. Note that both P4-Switches apply the same probing process, exchanging information asynchronously and independently.

## V. Evaluation

We have implemented the proposed method in P4 using the Tofino Native Architecture (TNA). The evaluation was carried out in our testbed using a Tofino-based P4-Switch (Stordis BF2556X-1T) and x86 servers running Codesys IDE, Codesys SoftPLC [7] and an IO device based on the open-source ProfiNet device stack of RT-Labs [8]. Note that we use a single P4-switch in the evaluation, implementing the two instances (two pipes) of the proposed method at the two sides of the radio link. The reliable radio link is emulated by a loop directly interconnecting two ports of the switch.

The scenario presented in Fig. 7 shows the individual ProfiNet IO data (*ProfinetAB*, *ProfinetBA*), CP-ACK (*ACKs*) and probe (*Probes*) packets on the radio link. *ProfinetBA* is used as reference, representing the data communication if the proposed traffic reduction method is not applied. In this case, switches only do simple forwarding. For comparison, both *ProfinetAB* and *ProfinetBA* apply the same sending frequency; an IO data packet is generated in every 8 ms. One can also observe that Probes are continuously present. These packets are sent by the P4-Switches to monitor the state of the radio link and to exchange information about the availability of devices deployed on the different sides of the critical link. Note that this traffic represents a constant overhead of the proposed method and is independent of the number of IO devices and PLCs in the system.

The timeline of the shown measurement is split into 7 regions: 1) In the first phase, the traffic reduction method is not applied at all. *ProfinetAB* packets are simply forwarded by the switches as for *ProfinetBA*. One can observe that there is no CP-ACK packets in the system. The packet rates are the same for *ProfinetAB* and *ProfinetBA*. 2) In the second phase, we turn the proposed traffic reduction method on. At the beginning of this interval, the switch at the other side of the radio link start storing the IO data and sending CP-ACK responses. At the end of the period, the other switch starts filtering out the unchanged *ProfinetBA* data packets and generating automatic replies without radio transmission. 3) The third phase illustrates the effect of our traffic reduction method, resulting in packet transmission over the radio link only if the packet data was changed. The data changes rarely, 6 times in this region. One can observe that for each transmitted IO data packet a CP-ACK packet is also transmitted, leading to two transmitted packets instead of one. However, even with this overhead the number of transmitted packets is negligible, compared to the reference traffic *ProfiNetBA*. 4) In the fourth phase, each packet carries new data bytes, illustrating the case when traffic reduction cannot be achieved, and, moreover, CP-ACKs double the overall traffic to be transmitted over the radio. Note that the proposed traffic reduction method can dynamically be turned on and off, and thus the traffic in this case can be maximized at the original level (same as *ProfiNetBA*). 5) In the next phase, the device turns back to an inactive state with few changes in the data. 6) We stop the traffic reduction method and switch back to normal forwarding of *ProfiNetAB* traffic. 7) The traffic reduction method is not applied as in the first phase.

One can observe in the previous measurement that the achievable traffic reduction capabilities depend on how frequently the IO data to be transmitted changes, but there are other factors as well: the frequency IO devices send data exchange packets with, and the response time of CP-ACK packets (the time between forwarding a new data packet through the radio link and receiving the corresponding CP-ACK as shown in Fig. 3). We generated a test traffic with various parameters to show the efficiency of our traffic reduction method. Fig. 8 depicts how the traffic of a single IO device can be reduced in average. The IO device sends data packets in every 8 ms. The probability of data change and the CP-ACK response times are varied. The dashed black line at 100% corresponds to the original traffic intensity when the proposed method is not applied. One can also observe that the CP-ACK response time affects the traffic reduction efficiency significantly, since this is the time needed for synchronizing the two P4-Switches. During this period, data packets cannot be filtered out and need to be forwarded through the radio link, also generating extra CP-ACK packets. In practical scenarios, the CP-ACK response time can be in the range of 1-10 ms, where traffic reduction has clear benefits if the data change probability is less than 0.3-0.5.

Note that for a single device we may generate more data than without applying the method, but in industrial environment we have to deal with the traffic of 100s or even 1000s of IO devices with various sending frequencies and data change probabilities, therefore the overall traffic reduction may still be beneficial. Let us consider the following example setup with 3 IO devices: a conveyor belt, a gluing device and a third one that puts a decoration into the glue. Only one of them is active at the same time. Supposing that they send data packets at the same rate, the proposed method can reduce the original $1+1+1=3$ unit of traffic to $1.5+0+0=1.5$ (without the constant overhead of probe traffic). The preliminary evaluation has mostly focused on the functional validation, and kept

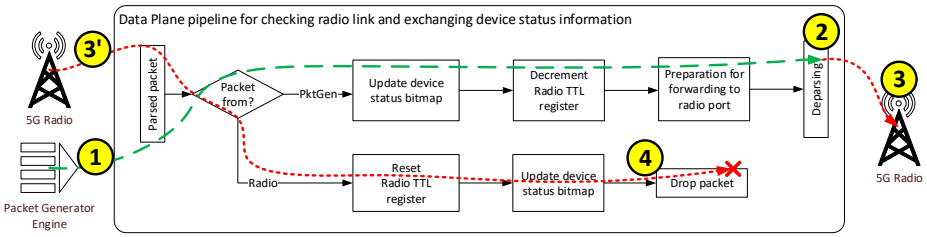


Fig. 6. Data plane pipeline for exchanging state information and monitoring the radio link.

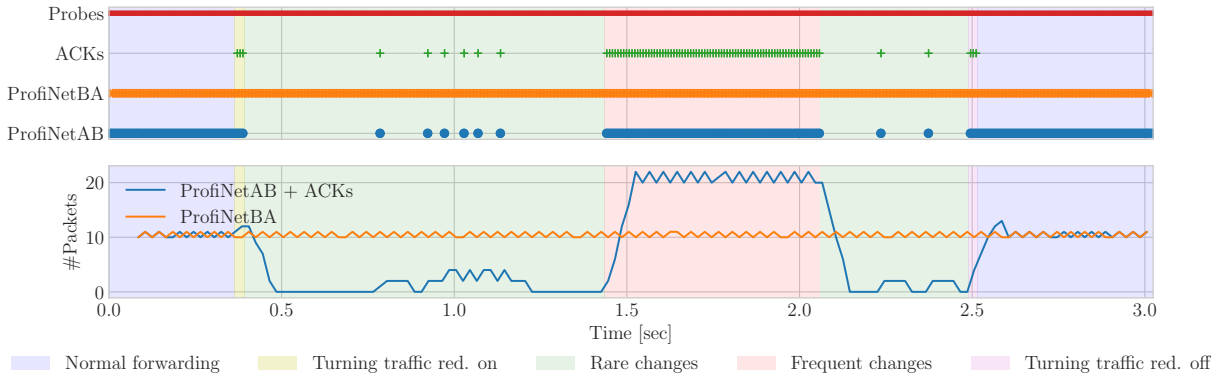


Fig. 7. Individual packets and the number of packets transmitted over the radio link (moving average with 85 ms window).

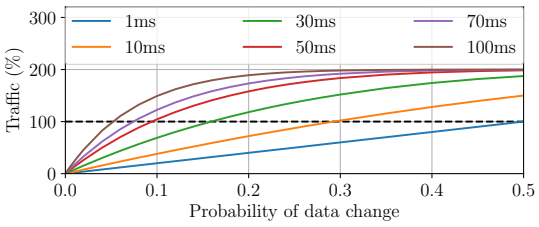


Fig. 8. Efficiency of traffic reduction for different CP-ACK response times

scalability and performance measurements for future works.

### A. Deployability and Limitations

The proposed solution does not require any changes in the other elements of the infrastructure. We imagine our final product as a plug-and-play enhancement to an existing system. The method can dynamically turned on and off, and thus it can only be applied for the traffic of selected devices. The use of hash functions for change detection raises some questions. The occurrence of false negatives is possible even if it is unlikely. However, we can compare the original values if the relevant part of the IO data is considerably small. In general, the size of ProfiNet IO data is between 40 and 1440 bytes. Our solution supports only the minimum sized packets currently.

## VI. Conclusion

Efficient PLC communication over the radio requires local management of device states. Using in-network computing, we presented a prototype that works with an existing industrial protocol. We have shown that our method does not affect the reliable operation and thus significantly reduces the average load on critical links (e.g., 5G radio). Though our preliminary results show the method's practical benefits, there are further aspects to consider and investigate as future work, including scalability, adaptability, robustness and performance.

## Acknowledgment

"Application Domain Specific Highly Reliable IT Solutions" project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

## References

[1] R. Pigan and M. Metter, *Automating with PROFINET: Industrial communication based on Industrial Ethernet*. John Wiley & Sons, 2008.
[2] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (sdn)," *Computer Networks*, vol. 112, pp. 279–293, 2017.
[3] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890
[4] J. W. Lockwood *et al.*, "Netfpga–an open platform for gigabit-rate network switching and routing," in *2007 IEEE Int. Conference on Microelectronic Systems Education (MSE'07)*, 2007, pp. 160–161.
[5] C. Kim *et al.*, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.
[6] S. Laki, P. Vörös, and F. Fejes, "Towards an aqm evaluation testbed with p4 and dpdk," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 148–150.
[7] "Codesys control for linux sl." [Online]. Available: https://store.codesys.com/codesys-control-for-linux-sl.html
[8] "Rt-labs p-net." [Online]. Available: https://github.com/rtlabs-com/p-net